SAM: Steady Affine Motions

Jarek Rossignac
School of Interactive Computing
College of Computing
Georgia Institute of Technology

Àlvar Vinacua
Department of Software
Technical University of Catalonia

November 23, 2009

Abstract

An affine motion is a continuous map from time value t to an affinity A_t. It is a SAM (Steady Affine Motion), when $A_t = A^t$. Although the beauty of a motion is subjective, the above equation provides one mathematical characterization and includes the screw ("universal instantaneous") motion and the golden ("mirabilis") spiral. Although a real matrix, A^t, may not exist, we show that it does for a dense set of affinities A covering a significant range of rotations and shears around the identity and that it may be computed efficiently and robustly in two and three dimensions using closed form expressions. SAMs have remarkable properties. For example, the velocity of any point remains constant, both in the global (fixed) and local (moving) frames, which facilitates the exact computation of derived entities, such as the envelope surfaces used to define the boundary of a swept volume. We say that a pattern of features F_i is steady when there exists an affinity M such that $F_i = M^i F_0$. Each M^i is a frame of a SAM and may be computed as $A^{\frac{i}{n}}$, where A is the affine relation $F_n = A F_0$ between the first and the last feature. This option makes it possible to edit directly the feature count n or the cumulative transformation A.

1 Introduction

We propose closed form expressions for computing Steady Affine Motions (SAM) that interpolate two poses related by an affinity.

1.1 Affinities

In this section, we review a few properties of affinities and start by introducing our terminology and notation. Affinities (affine transformations) of arbitrary dimension have been studied extensively in Mathematics. An affinity is a homeomorphism between $\mathbb{R}^n$ and itself and hence is invertible. It preserves co-linearity, parallelism, and ratios along a line, but in general, does not preserve angles or distances.

Here, we focus on orientation-preserving affinities in two and three dimensions. They comprise combinations of translations, rotations, and non-uniform scaling and are important in Computer-Aided Design, Computer Graphics, Animation, Computer Vision, and Robotics [Mor07, GDCV98].

In three dimensions, the image $A\,P$ of a point $P = (x, y, z)$ by an affinity A may be written as $T + x\vec{I} + y\vec{J} + z\vec{K}$, where T is a point and the vectors $\vec{I}$, $\vec{J}$ and $\vec{K}$ are linearly independent. We will also use $\vec{T}$ to denote the position vector of T: $\vec{T} = \overrightarrow{\mathbf{0}T}$.

$\vec{I}, \vec{J}$ and $\vec{K}$ are the images by A of the vectors of the universal orthogonal basis $\{(1,0,0),(0,1,0),(0,0,1)\}$, while T is the image by A of the origin $\mathbf{0} = (0,0,0)$. Hence, We represent A by the set $[\vec{I}\vec{J}\vec{K}T]$, which defines a local coordinate system and may be stored as a 3×4 matrix, or (for consistency with matrix multiplication tools) using homogeneous coordinates as a 4×4 matrix. Such matrices are used in constructive representations of solid models, in scene graphs defining graphics scenes or animations, and in display lists or rendering commands supported by graphic adapters. We will use the symbol L to denote the linear part of A, i.e., the 3×3 sub matrix $[\vec{I}\vec{J}\vec{K}]$. We can write $A\,P = L\,(\overrightarrow{\mathbf{0}P}) + T$, which displaces T by a linear transformation of vector $\overrightarrow{\mathbf{0}P}$.

A is orientation-preserving when the determinant of L is positive.

An affinity may be specified as the composition of parameterized primitive transformations (rotation, translation, scaling, shear) [Mor07], which are supported by most design systems. Its matrix representation may be computed using standard 4×4 matrix multiplication from the matrix representations of these primitive transformations.

An affinity may also be specified as the difference between two affinities B and C. For example, consider a shape S and the two instances of it $B\,S$ and $C\,S$. We want an affinity A that transforms $B\,S$ into $C\,S = A\,(B\,S)$. Hence, by the associativity of matrix multiplication $C = A\,B$ and hence $A = C\,B^{-1}$.

Finally, an affinity in $\mathbb{R}^n$ may be specified by $n+1$ pairs of matching points (S_i, F_i), that define $n+1$ point-equality constraints: $F_i = A\,S_i$ for i in $\{0, 1, \ldots, n\}$, each resulting in n equations, one per coordinate. One may compute the coefficients of A by solving the corresponding system of $(n+1) \cdot n$ linear equations. When more than $n+1$ pairs are given, least-squares registration [BLCD02] may be invoked to compute an optimal fit.

1.2 Steady Affine Motion (SAM)

In this subsection we introduce the notion of a Steady Affine Motion (abbreviated SAM). An affine motion (not to be confused with an affine transformation) is a continuous mapping from the scalar time parameter t to the set of affinities. Let A_t be such an affine motion. More precisely, A_t denotes the affinity associated with time value t.

Without loss of generality, we can shift and scale time so that the motion starts at time 0 (and hence A_0 is the identity $\mathbb{1}$) and so that at time $t = 1$, we obtain some desired affinity A (hence $A_1 = A$).

If we are given two affinities, B and C, and wish to compute an affine motion that progressively interpolates between them, we use, as explained above, $A = C\,B^{-1}$ and to each time t associate the affinity $A_t\,B$.

Note that an $\mathbb{1}$ and A or a B and C pair does not suffice to completely define the interpolation, since an infinity of affine motions exist that satisfy these constraints.

To remove this ambiguity, we propose to require (when possible) that the affine motion be steady, as defined below.

We define an affine motion A_t to be steady if and only if there exists an affinity A such that

$$A_t = A^t \tag{1}$$

We show that the instantaneous velocity of a point under a steady motion remains constant over time in both the local (moving) and in the global (fixed) frames. In fact, we propose to use the integration of acceleration (i.e., the change of velocity in the local frame) as a measure of unsteadiness.

Although elegant, this formula uses a non-standard notation of an affinity raised to a real power t. The equivalent notation

$$A^t = e^{t \log(A)} \tag{2}$$

is more explicit, but involves the log of a 3×4 matrix, unless one uses a homogeneous 4×4 representation of A, which requires computing the log of a 4×4 matrix, for which as far as we know there is no simple closed form.

The notion of using the exponential to parameterize a group of transformations dates back to the seminal work of Sophus Lie in the nineteenth century. For rotations, it has been promoted for graphical applications by Grassia [Gra98]. Alexa [Ale02] has used it also to compute linear combinations of homogeneous matrices. In fact, our equation (2) is consistent with Equation (7) in [Ale02], where the linear combination $(1-t)\mathbb{1} + tA$ is expressed as $exp((1-t)\log(\mathbb{1}) + t\log(A))$ which simplifies to $e^{t\log(A)}$ since $\log(\mathbb{1}) = 0$.

Our solution differs from Alexa's in two important ways:

1. When interpolating between B and C, Alexa proposes to use $e^{(1-t)\log(B)+t\log(C)}$ which is $e^{\log(B)+t(\log(C)-\log(B))}$ which is not equivalent to the expression we use, namely $e^{t\log(C\,B^{-1})}\,B$, except when B and C commute. Note that when B and C do not commute (which is the general case), the solution proposed in [Ale02] is not steady.

   For example, consider the case where $B = Rot(90°)$, and $C = T(6,0)$. Lets consider the transformation X associated with $M_{\frac{1}{2}}$. That is an X such that $M_{\frac{1}{2}} = X\,B$. Then steadiness provides that $X^2\,B = C$ (and in this sense $M_{\frac{1}{2}}$ is mid-way between the two). Using the formulation in [Ale02] yields

$$X_{Alexa} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{6\sqrt{2}}{\pi} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{6(\sqrt{2}-2)}{\pi} \\ 0 & 0 & 1 \end{bmatrix}$$

   and SAM yields

$$X_{SAM} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 3 \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 3\sqrt{2}-3 \\ 0 & 0 & 1 \end{bmatrix}.$$

   While they both turn the model by the same amount, $X_{SAM}^2\,B = C$, but $X_{Alexa}^2\,B$ is a translation by only $\frac{12\sqrt{2}}{\pi} \approx 5.402$ in the x direction.

2. Instead of computing $\log(A)$ for a 4×4 matrix A using dimension independent, iterative, numeric tools, we propose here a new approach that is limited to two and three dimensions, but performs the computation using closed form expressions directly, without iterations. This approach has robustness and performance benefits.

Furthermore, the concept of steadiness introduced here extends the concept of rigidity in two ways:

1. When the evolving feature $F(t)$ remains congruent to F_0 for all t, the motion may be considered as rigid, regardless of the motion (i.e., how the position and orientation of $F(t)$ evolve over time). In contrast, requiring steadiness defines the motion completely.

2. When the shape of the feature $F(t)$ deforms over time, the concept of steadiness provides a formal and usable definition of the most rigid deformation.

In our approach, we separate the translational part T from the linear part L: recall that $A\,P = L\,(\overrightarrow{\mathbf{0}P}) + T$. In the general case, A has a unique fixed point Q (singular cases when it does not, require custom treatment as discussed in Section 3). Then, $A\,Q = L\,(\overrightarrow{\mathbf{0}Q}) + T$ yields , $Q = L\,(\overrightarrow{\mathbf{0}Q}) + T$. Subtracting the terms of these two equations, one obtains $A\,P - Q = L\,(\overrightarrow{\mathbf{0}P}) + T - L\,(\overrightarrow{\mathbf{0}Q}) - T$, which simplifies to $A\,P = L\,(\overrightarrow{QP}) + Q$, so $A\,P - A\,Q = L\,(\overrightarrow{QP})$, since $A\,Q = Q$. We therefore have $A\,(\overrightarrow{QP}) = L\,(\overrightarrow{QP})$, and hence $A\,P = Q + L\,(\overrightarrow{QP})$. For simplicity of exposition, we will usually refer to a coordinate system with origin at Q.

Hence, given the fixed point Q, we express the effect of SAM as

$$A^t\,P = e^{t log(L)}\,(\overrightarrow{QP}) + Q \tag{3}$$

The Extraction of Affinity Roots (EAR) introduced in this paper provides a closed-form expressions for A^t and for the fixed point Q when they exist. EAR also provides alternative closed-form expressions for $A^t\,P$ in the singular situations where Q does not exist or is not unique, or alternative (unsteady) solutions when the $\log(L)$ is not real.

1.3 SAM benefits

Before we discuss the benefits of using SAM, let us contrast it with other (unsteady) affine motions using a two-dimensional example. Figure 1 compares five affine motions that interpolate the same pair of control frames (affinities) using a triangle (top) and a rectangular image (bottom).

The columns correspond (from left to right) to (a) linear interpolation, (b) a modified version of ARAP ([ACOL00]) where a constant-speed translation along the line joining the barycenters has been added, (c) a linear interpolation between a forward and a backward logarithmic spiral —that interpolate the positions, orientations and uniform scaling , (d) a blending (weighted average) of three logarithmic spirals computed from the three pairs of sides of the triangles, and (e) the SAM presented in this paper.

For each morph, we report a measure of unsteadiness (which is zero for SAM). We formulate this measure as the integral over space and time of the acceleration in the

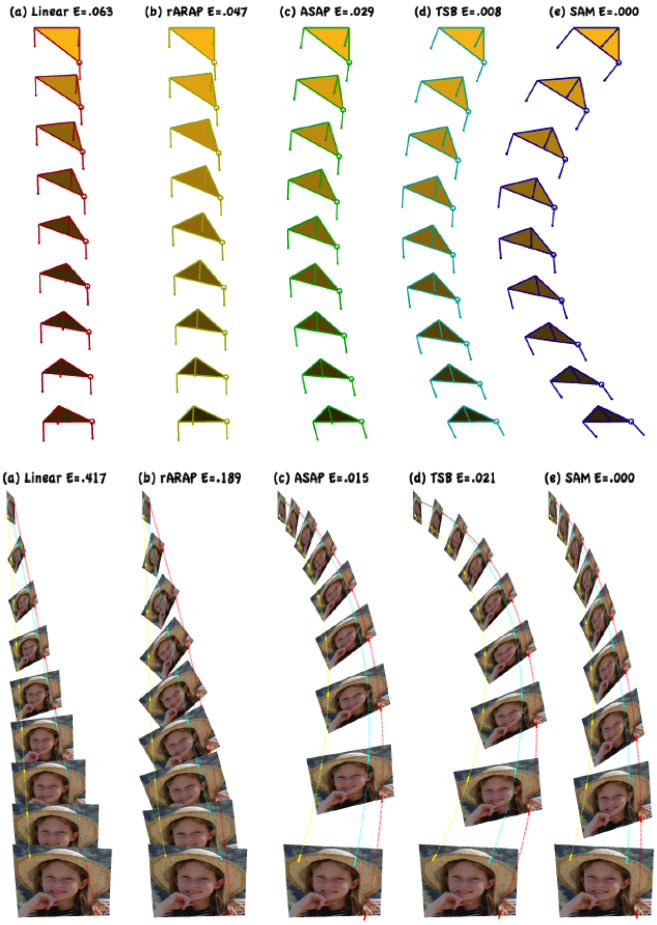

Figure 1: A comparison of different affine motions acting on triangles and textured quads.

moving frame. We compute it using a dense regular sampling of a vicinity of the moving shape.

Comparative evaluation of the esthetic advantages of each option would be —by definition— subjective and, in any case, better served by viewing continuous motions rather than the sparse intermediate frames shown in Figure 1. Furthermore, the best scheme to use will also depend on the application. Nevertheless, we would like to highlight some of the properties of SAM displayed in this figure. In the top picture, notice that the velocity vectors shown at the vertices stay fixed with respect to the deforming triangle only in the case of SAM. In the bottom picture, notice how SAM achieves the most uniformly spaced and least distorted pictures. More precisely, SAM maintains a constant affine relation between consecutive frames, while the other motions do not.

Using a SAM as an interpolating affine motion has several benefits:

1. The resulting motion is coordinate invariant (i.e., it is not affected by a change of coordinate system). The importance of this property has been stressed by several authors [RK01, LS02, KCŽO08]

2. A SAM is direction independent (the reverse motion is obtained by swapping the initial and final affinities), since $A_t = A_{(1-t)}^{-1} A$, which is advocated as important in [KCŽO08]

3. A SAM ensures that integral properties (volumes, area) evolve monotonically during the motion (see Section 5.4).

4. When interpolating rigid body transformations, the resulting motion is a screw and hence preserves rigidity (which is often desired in animation [ACOL00]) and satisfies the shortest path (smallest rotation angle) and constant speed (linear variation of rotation angle and translation distance) properties advocated in [KCŽO08].

5. A SAM motion of a point has constant speed (i.e., the velocity of the point remains constant through time in both the local and the global frame, which simplifies the computation of swept region envelopes as discussed in more detail in Section 5.5 (see [RKS+07]). We show in Section 5.1 that the only affine motions with this property are those defined by equation (1)).

6. The motion followed by a point combines a logarithmic spiral in a plane with an exponential scaling in a direction transversal to the plane. The simplicity of the mathematical description of this motion facilitates the computation of point/plane collisions, hence suggesting possible extensions of the collision prediction techniques originally developed for screw motions [KR03] to affine motions.

7. SAM produces visually pleasing animations where points travel along naturally curved arcs without unexpected inflections. Such curved trajectories are often preferred to linear motions by artists [JT95].

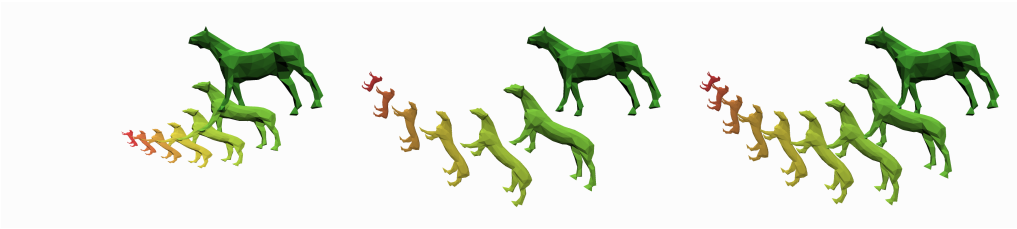

Figure 2: (Left) Original steady pattern from F_0 (green) to F_n (red). (Center) New steady pattern for a different $F_n = A\,F_0$. (Right) The same control frames F_0 and F_n, but a different copy count n.

8. Industrial designers, architects, artists, and other users of CAD or animation systems occasionally wish to produce steady patterns [JR09, vERR93, WWSR03, PBPS99, LG06, AIS77] of shape or geometric features, F_i, typically specified by

an initial feature F_0 and by the first copy $F_1 = M\,F_0$, where M is an affinity defined by the designer. The subsequent copies are defined by $F_{i+1} = M\,F_i$, and therefore are regularly sampled discrete frames, $F_i = M^i\,F_0$. Let A be the affinity such that $F_n = A\,F_0$. Hence $M = A^{\frac{1}{n}}$ and $F_i = (A^{\frac{1}{n}})^i\,F_0$ which we write as $F_i = A^{\frac{i}{n}}\,F_0$. It is often less tedious to edit F_n directly (by tweaking A) rather than by tweaking M. The EAR algorithm proposed here may be used to compute $A^{\frac{i}{n}}$. Furthermore, the designer may wish to keep F_0 and F_n fixed, but change the count n of copies. Again, the new steady pattern may be computed using EAR (Figure 2).

1.4 Organization of the paper

The remainder of the paper is structured as follows. We first discuss prior art on rigid and affine motions (Section 2). We then present the derivation of the closed-form Extraction of Affinity Roots (EAR) introduced in this paper for computing A^t in two and in three dimensions, when it exists (Section 3). We then discuss the existence of SAM and report experimental results on charting the space for which a solution (real matrix A^t) exists (Section 4). We finally present some properties of SAM and point out some of its potential applications to modeling and animation (Section 5).

2 Prior art

We split prior art in three groups: (1) the special case of screw motions, (2) previously proposed numeric solutions for extracting roots and logarithms of linear transformations, and (3) unsteady alternatives.

2.1 Screw motions

Since we assume that L (the linear part of affinity A) is orientation preserving, when the column vectors of L form an orthonormal basis, A represents a rigid body transformation (Euclidean transformation) and may be expressed as a rotation (defined by the matrix L) and a translation (from the origin to point T).

In two dimensions, the corresponding SAM is either a pure translation (when $L = \mathbb{1}$) or a pure rotation around a fixed point Q that may be easily computed by solving the linear system of two equations $A\,Q = Q$. The angle a of the rotation is defined by the first column vector of L: $I = (\cos(a), \sin(a))$.

In three dimensions, the corresponding SAM is a screw motion [RK01] (Fig. 3), which combines a constant angular velocity rotation (by angle ta) around a fixed axis (with direction D and passing through the fixed point Q) and a constant velocity translation (by distance td) in the direction D.

Simple closed-form expressions have been proposed for computing the screw parameters (D, Q, a, and d) using variations of the Rodrigues formula [RK01, LKG+03].

When applied to a rigid body motion, A, the more general EAR approach proposed here computes D as the eigenvector of the unique real eigenvalue of L and Q as the fixed point of A.

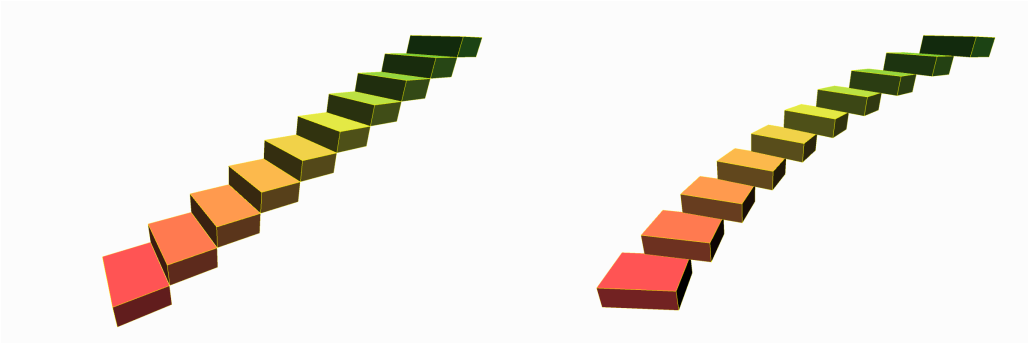

Figure 3: A stair case generated by a pure translation (left) and by a screw (right).

The screw motion may also be computed and animated using dual quaternions [Cli82], which offer benefits for GPU support and for computing weighted averages of more than two transformations. For example, the Screw Linear Interpolation (ScLERP) proposed in [KCŽO08] as a generalization of SLERP [Sho85] produces screw motions.

2.2 Numeric solutions

We discuss here previously proposed iterative approaches for computing $log(L)$ upon which the general case computation of A^t is formulated (see Equation (3)). The discussion of the existence of a real solution is deferred to Section 4.

There is abundant bibliography on the computation of exponentials, logarithms and square roots of matrices of arbitrary dimension. See for example [Mei05, CHKL00, GV96, DH03] or [Hig86]. The reported algorithms are also available on standard platforms, such as Matlab [HH05].

For example, a successful approach was reported by Alexa [Ale02], who approximates $log(L)$ by a series expansion. Since such an approach only converges for matrices close to the identity, Alexa first computes an approximation of L^t by performing a series of square root extractions during a binary partition of the unit time interval. Because of the binary search and the series expansion, this approach is slower than the EAR proposed here. Furthermore, to extract the square roots, Alexa relies on the Denman-Beavers iteration, which was found to be unstable [Hig86].

Another approach [DH03] is based on the Shur decomposition, which performs a change of basis to obtain an upper-triangular matrix. Since the change of basis commutes with log and exponentiation, the problem is reduced to the simpler problem of computing the log of an upper-triangular matrix.

As pointed out in [KCŽO08] these general, dimension-independent, solutions suffer from numeric errors. Furthermore, their robust implementation is delicate because of possible convergence and branching problems. For example, consider the two dimensional matrix $A = \begin{bmatrix} -2 & 0 \\ 0 & 2 \end{bmatrix}$ which has a double negative eigenvalue. The log functions of both Matlab and Octave —a public domain clone of Matlab (see [Eat00])— return a complex

solution, when in fact a real solution exists: $\begin{bmatrix} \log(2) & -\pi \\ \pi & \log(2) \end{bmatrix}$ and is correctly computed by our EAR solution.

Hence, to overcome these problems, we offer a closed-form solution for computing the real log of 2×2 and 3×3 matrices when it exists.

2.3 Unsteady alternatives

In this subsection we discuss previously proposed interpolation techniques that produce unsteady motions.

We split them into three categories: (1) Those dealing with rigid body transformations. (2) Those dealing only with the linear sub-matrix L, hence ignoring the translation part of the motion (such an approach is appropriate when only the shape and orientation of the moving object are important or when its center of mass must follow a trajectory prescribed by physics or artistic concerns). And (3) those dealing with more general affinities.

2.3.1 Rigid body transformations

Several authors have proposed techniques for blending spherical spline curves on the S^3 unit sphere [BF01b, Sho85, Sho87, Duf86, WJ93, RBG88, KN95]. Barr et al. [BCGH92] also use quaternions, but compute a smooth rotation that interpolates a given sequence of constraint poses. Kim et al. [KKS95] generalize this approach by providing a general framework for interpolating motions with unit quaternion splines. Ma et al. [MCJ00] used B-splines to compute smooth interpolating rigid body motions. They first approximate the reference frames by B-splines that do not preserve orthogonality, and then compute the best orthogonal approximations through a Newton iteration (for each desired time t_i).

Kavan et al. [KCŽO08] compare their (ScLERP) approach to two other unsteady solutions: DLB and DIB. Their Dual quaternion Linear Blending (DLB) is a generalization of the QLB [Kv05], which performs a linear interpolation of quaternions, followed by a normalization. DLB uses the same fixed screw axis as ScLERP, but changes the rotation angle and translation distance non-linearly and differently. The discrepancy is small. The Dual quaternion Iterative Blending (DIB) is a generalization of spherical averages [BF01a].

2.3.2 Interpolations of L

Techniques described in this subsection focus only on the linear part, L, of the transformation, hence ignore translation. Note that they may be combined with position interpolation techniques, such as uniform speed or logarithmic spirals (see Figure 1 c).

The linear interpolation of the coefficients of L is often not acceptable because it may temporarily invert the shape, and may evolve the area/volume in a non-monotonic fashion.

Shoemake [Sho85] and Shoemake and Duff [SD92] use spherical linear interpolation (SLERP) to interpolate orientations represented using quaternions and combine it with a linear interpolation of the coefficients of a stretch matrix.

The As-Rigid-As-Possible interpolation [ACOL00] combines an SVD decomposition with a linear interpolation of the remaining shear. In two dimensions, let $A = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}$ represent $A_1 A_0^{-1}$, and $L = \begin{bmatrix} a & b \\ d & e \end{bmatrix}$. A singular value decomposition (SVD) is used in [ACOL00] to express L as the product $R(\alpha)\, Y\, R(\beta)$, where $R(\alpha)$ and $R(\beta)$ are orthogonal matrices and where Y is diagonal.

The SVD returns two orthogonal matrices U and V, and the principal values s_1 and s_2. These matrices will be pure rotations when their determinant is positive (i.e. when it is 1). SVD may return matrices with negative determinants, but in this case, and if the affinity is orientation-preserving, exchanging the variables will yield orthogonal matrices with positive determinant, from which we can then extract the angles (α and β). The affinity can be written as $L = R(\alpha)\, R(\beta)\, (R(-\beta)\, Y\, R(\beta)) = R(\alpha + \beta)\, B$, where $B = R(-\beta)\, Y\, R(\beta)$ is a shear matrix.

Then, a morph of L may be formulated as $L(t) = R(t(\alpha + \beta))\, ((1 - t) \cdot I + t \cdot B)$. The linear part of the affinity at time t is obtained as $L(t)\, L_0$. The translational part $O(t)$ is not defined in this approach.

Notice that since α and β are computed independently, their sum's absolute value may exceed 2π, in which case the authors of ARAP recommend adding/subtracting 2π (see also [FTA05, CS03] for more details).

Several approaches have been proposed for constructing morphs that minimize distortion (or equivalently maximize rigidity).

Shoemake and Duff [SD92] used polar decomposition to parameterize and animate an affinity. They decompose an affinity A into the product $R\, S$ of a rigid motion R and a symmetric positive-definite stretch matrix S. R is computed by minimizing the Frobenius norm $||A - R||$. Then, both R and S are parameterized by time and the animated affinity composed as $R(t)\, S(t)$. $R(t)$ is animated as a screw and $S(t)$ is animated using a linear interpolation of the matrix coefficients. Shoemake and Duff apply their approach to construct smooth motions that interpolate more than two affine transformations. Computing independent splines that interpolate the coefficients of the matrices of these transformations may produce unwanted distortions because the orthogonality of $R(t)$ and the rigidity of the $S(t)$ are not enforced.

Hyun et al. [HJK02] propose an approach for progressively refining an affine spline motion using an iterative procedure that involves knot insertion and degree elevation and operates on a curve in a linear 12-dimensional space (of the affine matrix coefficients), where they relate the fairness of the curve in 12-space to the rigidity and fairness of the affine motion in 3-space. Hence, their rigidity measure corresponds to non-quadratic fairness measures, and requires a numerical approximation. They measure the rigidity by integrating over time the first or second derivatives of the lengths of a certain set of witness vectors as they are transformed (see their equations (12) and (13)). Because of the non-linear nature of the problem, the solution requires an iterative minimization. In cases where the final pose is the image under a rigid motion of the original, EAR will make all the witness vectors constant in length, and therefore both measures will be minimized (and will actually be zero). However, in other more general cases, the measure proposed in [HJK02] will depend on the choice of witness vectors. Changing those will likely yield a

different solution. EAR yields a unique solution, the steady one, and therefore is different from [HJK02].

As previously discussed, Alexa et al. (see [ACOL00]) use a decomposition similar to the polar decomposition in [SD92], but instead of finding the rotation through a minimization, they resort to the SVD (Singular Value Decomposition, see [GV96]) of the matrix of the linear part to compute a canonical rotation. They do not offer a formal definition of rigidity.

In their Dual quaternion Iterative Blending (DIB), Kavan et al. [KCŽO08] propose an iterative extension of Alexa's 'log-matrix blending' ([Ale02]) that makes a constant-speed, shortest path solution, but warn that the matrix exponential and logarithm routines it uses require numeric solutions to be applied iteratively, which reduces performance and accumulates numeric errors.

3 Extraction of Affinity Roots (EAR)

In this section, we provide the details and justifications of our EAR algorithm, which computes A^t, when it exists as a real matrix, or an unsteady alternative of it, when it does not. In the next section, we discuss the precise mathematical characterization of when a real A^t exists, and our experimental findings regarding the compactness and extent of a useful safe region in the configuration space of L for which a real solution A^t exists. We distinguish, below, general situations, special cases, and situations where no SAM exists.

The **general solution** in three dimension falls into one of two possible cases:

Case 1: L has three real eigenvalues, all are different from 1, and the space W spanned by the eigenvectors has dimension 3. In this case, we use closed form expressions to perform a change of basis that diagonalizes L and perform exponentiation on the diagonal terms. Instances corresponding to this case are painted red in the example shown in Figure 4.

Case 2: L has a single real eigenvalue that is different from 1 (and two conjugate imaginary eigenvalues). In that case, we cannot resort to diagonalization. Instead, we build the solution as a composition of a two-dimensional solution (for which we provide closed form expressions) with an exponential scaling in the complementary dimension. Instances corresponding to this case are painted blue in Figure 4.

A closed form solution also exists in several **singular configurations** that are identified by the number of real eigenvalues that are equal to 1, the presence of algebraically multiple eigenvalues, and by the dimension of W (see Sections 3.3 and 3.4).

Finally, in some configurations, **no real solution exists** (as discussed in Section 4). In these cases, we provide a closed form expression for computing an unsteady alternative that extends our steady construction.

The robustness of our EAR algorithm is illustrated also in Figure 5, where as in Figure 4, we first compute a steady pattern of affinities B_i (bottom row) and a steady pattern of corresponding affinities C_i (top row). Then, for each pair (B_i, C_i), we compute their SAM and display it as a steady pattern of features F_{ij}. As the operator changes

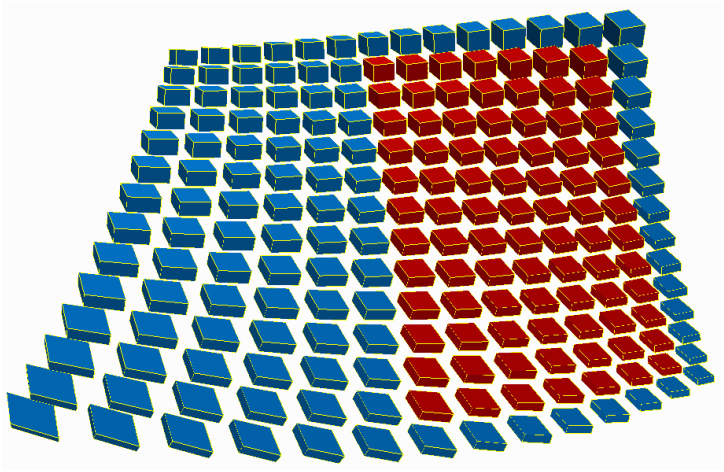

Figure 4: A bidimensional arrangement of features constructed by computing SAMs along the top and bottom rows and then interpolating SAMs along the columns. The color coding distinguishes the two general (and most common) cases.

interactively the affinities applied to the four corner shapes, the bidirectional matrix of shapes —recomputed in realtime— exhibits a smooth behavior throughout the bidirectional pattern and through time.

To simplify exposition, we structure our presentation and justification bottom up, starting with primitive tools in two dimensions, and then scaffolding to EAR implementations in two and then in three dimensions. We start by providing the pseudocode and rationale for computing the real log of a 2×2 matrix, U, when it exists, or an alternative, when it does not. Then, we provide the pseudocode and rationale for computing the exponential A^t in two dimensions, given L and a fixed point Q. Then, we provide the pseudocode and rationale for computing A^t in two dimensions. Finally, we explain how to use these tools for computing A^t in three dimensions.

3.1 Log of a 2×2 matrix U

Given a 2×2 matrix $U = [u_{ij}]$, we use algorithm 1 to compute the real matrix $\log(U)$ or its substitute (when no real log exists).

The details and justification of our treatment of each case are provided in the following subsections. We include a summary here.

In situations classified as `case01`, we invert the exponential given by equation (5) bellow, following the steps delineated in the discussion in the derivation of equations (6) through (12) and immediately after them. We distinguish the case where `D<0`, which implies we have two conjugate eigenvalues of U, and the equations may be solved by substituting $Y = \theta i$ and solving for θ. When `D<0` is false, we have two real roots. We distinguish the case where they are positive (`case02`) in which case we may apply (12) directly, or when they are negative (the case of mixed signs is excluded by the orientation-

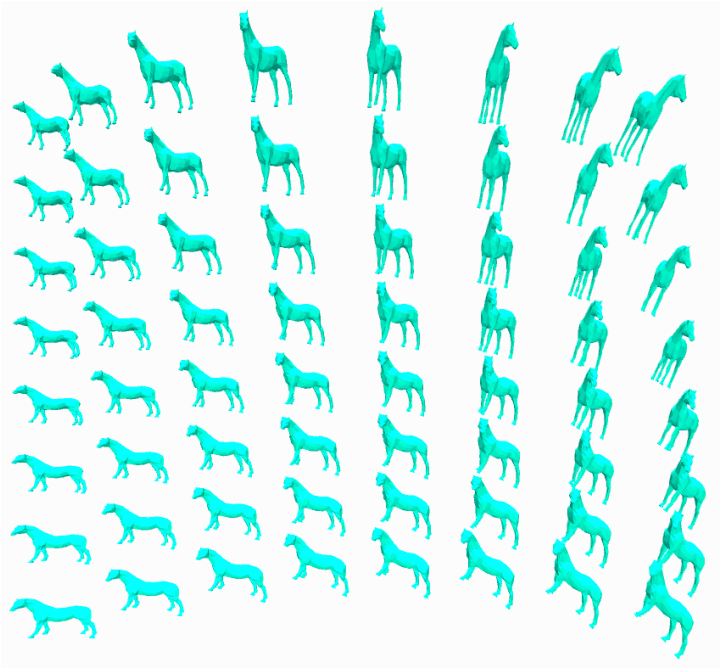

Figure 5: Another bidimensional arrangement of features.

preserving assumption). In the case of negative roots we need to discern if the condition for existence of the log (`(D==0)&&(u12==0)&&(u21==0)`) is met (see Section 4.1), or if we need to substitute an unsteady approximation (`case04`) since there is no real logarithm of the given matrix U.

3.2 Exponential of an affinity in two dimensions given L and Q

Given a 2×2 matrix L, a fixed point Q, and a time value t, we compute the real 2×3 matrix A^t when it exists, or a substitute when it does not using algorithm 2.

This code distinguishes cases in order to evaluate formula (5) below without resorting to complex arithmetic. This is discussed below after equation (5). The two general cases are dealt with by `case05` (when there are two conjugate eigenvalues) and by `case08` (when there are two positive eigenvalues). The case of negative eigenvalues is further divided into the only case where the real logarithm is defined (`case06`) and when we are using our surrogate approximation (`case07`).

3.3 Exponential of an affinity in two dimensions

Given a 2×3 affinity matrix A and a time value t, we compute the real matrix A^t when it exists, or a substitute when it does not using algorithm 3.

When 1 is not an eigenvalue of L, there is a unique solution to $AQ = Q$, and therefore

Algorithm 1 Find the logarithm of a 2×2 matrix

```
log2D(U) {
  D=(tr(U))**2/4-det(U);
  if (D<0) return case01(); // general case: complex conjugate e.v.
  else
    if (tr(U)>0) return case02(); // general case: two real, positive e.v.
    else
      if ((D==0)&&(u12==0)&&(u21==0))
           return case03(); // two identical negative e.v.
                            // .. with independent e.vectors
      else return case04(); } // unsteady substitute
```

Algorithm 2 The exponential of a 2×2 matrix multiplied by a scalar

```
Exp2D(L,Q,t) {
  B = log2D(L);               //Compute C_1,...,C_4 in (4)
  D=(tr(B))**2/4-det(B);
  if (D<-epsilon) return case05() // general case, conjugate roots.
                                  // ... epsilon=0.00001
  else
    if (both eigenvalues are negative)
      if (double root && W==2) return case06()
      else return case07() // substitute non-steady solution
    else return case08();} //general case, real positive roots.
```

a fixed point. Writing the affinity in a basis that uses that fixed point as origin of coordinates yields an affinity without a translation part (purely linear), and we can simply use the exponential discussed in the previous section.

The remaining possibilities are the identity (if 1 is a double eigenvalue of geometric multiplicity two), a pure shear (that has a whole line of fixed points) or a scaling in only one direction. The latter is simply handled in `case09` by exponentiation of the non-unitary eigenvalue in a diagonalized form. For the pure shear, `case10` uses the fact that

$$e^{t\begin{bmatrix} 0 & a \\ 0 & 0 \end{bmatrix}} = \begin{bmatrix} 1 & ta \\ 0 & 1 \end{bmatrix},$$

which is easily obtained using the series expansion of the exponential.

3.4 Exponential of an affinity in three dimensions

Given a 3×4 affinity matrix A and a time value t, we compute the real matrix A^t when it exists, or a substitute when it does not using algorithm 4.

Algorithm 3 The exponential of an affinity in the plane

```
Exp2D(A,t) {
  if (1 is not an eigenvalue of L) {
    compute fixed point Q;
    return Exp2D(L,Q,t);
  }else{
    c=algebraic multiplicity of eigenvalue 1.
    if (c==1) return case09()
    else if (span of eigenspace has dimension 2) return 𝕀
         else return case10();
  }
}
```

Again this code is justified in a following section, specifically in Section3.6. The different cases that need to be distinguished, are whether there are three real eigenvalues (`r==3`, cases 11 through 16) or just one (`else` part, cases 17 and 18). Within these two main categories, we deal differently with the problem depending on the number of real eigenvalues exactly equal to one. The reason is that when one is an eigenvalue of the linear part, there is no unique fixed point for the affinity, so that the translation part has to be dealt with, as discussed for cases B and C in Section 3.6. Notice however that these cases (11, 12, 13 and 17) are exceptional cases; the general cases are handled by `case14()` when there are three real eigenvalues with distinct eigenvectors, and by `case18()`, when there is only one real eigenvalue and it is not one. One last case distinction is needed when there are three real eigenvalues, for they may not have three linearly independent eigenvectors (for example, in case of a sheer). `case15()` and `case16()` are implemented for those cases using the much simpler expressions for the exponential in these cases:

$$exp\left(\begin{bmatrix}\lambda & 0 & 0\\ 0 & \mu & a\\ 0 & 0 & \mu\end{bmatrix}\right) = \begin{bmatrix}e^{\lambda} & 0 & 0\\ 0 & e^{\mu} & ae^{\mu}\\ 0 & 0 & e^{\mu}\end{bmatrix}$$

and

$$\exp\left(\begin{bmatrix}\mu & a & 0\\ 0 & \mu & b\\ 0 & 0 & \mu\end{bmatrix}\right) = \begin{bmatrix}e^{\mu} & ae^{\mu} & \frac{ab}{2}e^{\mu}\\ 0 & e^{\mu} & be^{\mu}\\ 0 & 0 & e^{\mu}\end{bmatrix}.$$

3.5 The mathematics behind the code

The proposed solution requires that we compute $log(U)$ for any 2×2 matrix U. We use the fact that $U^t = exp(t \cdot log(U))$, by the standard properties of the exponential.

Let $L = [a_{ij}]$ be the linear part of a two-dimensional affinity. We need to compute a matrix

$$C = \begin{bmatrix}C_1 & C_2\\ C_3 & C_4\end{bmatrix} = log(L) \tag{4}$$

Algorithm 4 The exponential of an affinity in three dimensions

```
exp3D(A,t){
  {ei} = eigenvalues of L;
  {Vi} = corresponding eigenvectors;
  W = dimension of the space spanned by the eigenvectors
  r=number of real eigenvalues ; // imaginary part < 0.000001
  if(r==3) {
    c=number of eigenvalues equal to 1;
    if (c==3) case11()
    else if (c==2) case12()
    else if (c==1) case13() // calls exp2D
    else
      if (W==3) case14()  // general case 1 (red)
      else if (W==2) case15()
      else case16();
  }
  else
    if (e1==1) case17();
    else case18(); // general case 2 (blue), calls exp2D
}
```

or in other words, a matrix C such that $e^C = L$.

The —possibly complex— eigenvalues of C are

$$\lambda_{1,2} = \frac{C_1 + C_4}{2} \pm \sqrt{\left(\frac{C_1 + C_4}{2}\right)^2 - (C_1C_4 - C_2C_3)}.$$

Let $D = \left(\frac{C_1+C_4}{2}\right)^2 - (C_1C_4 - C_2C_3)$ and $z = (C_1 - C_4)/2$. D is the discriminant of the characteristic polynomial, so its sign determines the nature (real or complex) of the eigenvalues, and their algebraic multiplicity. We use D everywhere to refer to this quantity, both in this section and in the algorithms for the logarithm and the exponential matrices (although in each case it may refer to the discriminant of a different matrix). We may form a matrix whose columns are linearly independent eigenvectors of C:

$$V = \begin{bmatrix} \frac{C_2}{\sqrt{D}-z} & \frac{-C_2}{\sqrt{D}+z} \\ 1 & 1 \end{bmatrix}.$$

Then $\begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} = V^{-1} \cdot C \cdot V$, and therefore one can compute the exponential of C as

$$\begin{aligned} e^{tC} &= V \cdot \begin{bmatrix} e^{\lambda_1} & 0 \\ 0 & e^{\lambda_2} \end{bmatrix} \cdot V^{-1} = \\ &\frac{1}{2} \begin{bmatrix} \frac{(\sqrt{D}+z)e^{\lambda_1 t}+(\sqrt{D}-z)e^{\lambda_2 t}}{\sqrt{D}} & \frac{C_2}{\sqrt{D}}\left(e^{\lambda_1 t} - e^{\lambda_2 t}\right) \\ \frac{C_3}{\sqrt{D}}\left(e^{\lambda_1 t} - e^{\lambda_2 t}\right) & \frac{(\sqrt{D}-z)e^{\lambda_1 t}+(\sqrt{D}+z)e^{\lambda_2 t}}{\sqrt{D}} \end{bmatrix}. \end{aligned} \tag{5}$$

This is the expression used in our code for the exponential. Notice that for real C_i — although it may involve complex numbers— this expression yields a real result: when $D < 0$, λ_1 and λ_2 are complex conjugates, so the sum of their exponentials is real, and the difference is purely imaginary. For the implementation we distinguish cases `case05` and `case08` to keep the computation completely within the real domain.

While we used general complex eigenvectors to obtain this formula, one can check that in the special cases (when C_2 and/or C_3 are zero) the resulting formula is still correct.

Let's further denote by $x = (C_1 + C_4)/2$, and $Y = \sqrt{D} = \sqrt{x^2 - det(C)}$. Letting $t = 1$ in the previous expression for the exponential, equating it to L, and operating, we obtain

$$z = \frac{(a_{11} - a_{22})C_2}{2a_{12}}, \tag{6}$$

and

$$C_3 = \frac{C_2 a_{21}}{a_{12}}. \tag{7}$$

Furthermore, summing the equations for a_{11} and for a_{22} and simplifying we find that

$$e^x = \frac{a_{11} + a_{22}}{e^Y + e^{-Y}}. \tag{8}$$

Next we notice that $Y^2 = z^2 + C_2 C_3$, and substituting z and C_3 using (6) and (7), we get that

$$C_2 = \frac{2a_{12}Y}{\sqrt{a_{11}^2 - 2a_{11}a_{22} + a_{22}^2 + 4a_{21}a_{12}}}. \tag{9}$$

Finally, using (8) and (9) to substitute in the equation for a_{12}, we obtain an equation that links Y with the entries in A only:

$$1 = \frac{(a_{11} + a_{22})(e^Y - e^{-Y})}{\sqrt{a_{11}^2 - 2a_{11}a_{22} + a_{22}^2 + 4a_{21}a_{12}}(e^Y + e^{-Y})}. \tag{10}$$

To solve this equation, we need to distinguish two cases. First, consider the case where the formula under the root is negative (handled by `case01` in the code above). This corresponds to the affinity having two complex conjugate roots. In this case, setting $Y = \theta i$, we obtain

$$\tan(\theta) = \frac{\sqrt{det(L) - \left(\frac{\text{trace}(L)}{2}\right)^2}}{\frac{\text{trace}(L)}{2}}. \tag{11}$$

and substituting backwards $Y = i\theta$ in equations (9), (7), (8) and (6), we find values for x, z, C_2 and C_3, and then obtain the final entries using that $C_1 = x + z$ and $C_4 = x - z$.

In the case when L has two positive eigenvalues (`case02` above), the expression under the square root in (10) is positive, and we may directly obtain, after writing the formula in terms of e^{2Y} only,

$$Y = \frac{log\left(\frac{\chi+1}{\chi-1}\right)}{2}, \quad \text{where } \chi = \frac{\frac{\text{trace}(L)}{2}}{\sqrt{\left(\frac{\text{trace}(L)}{2}\right)^2 - det(L)}}, \tag{12}$$

and again substitute in the previous equations (9), (7), (8) and (6) to get all the C_i.

Since we deal only with orientation-preserving transformations, the only remaining case is when L has two negative eigenvalues. If these are identical, and there exist two linearly independent eigenvectors, L is a rotation by π radians (plus perhaps a uniform scaling) and the logarithm $\begin{bmatrix} \log(-s) & -\pi \\ \pi & \log(-s) \end{bmatrix}$ —where s is the scaling factor— is readily available (`case03` and `case06`). In any other case with negative eigenvalues, the logarithm does not exist, nor does it make sense to speak of A^t for any non-integer t. In this latter case, there is no SAM. Nevertheless we propose a substitute solution that yields an unsteady motion. To do so, we operate formally, after noticing that $-L$ will have two real, positive eigenvalues, and we therefore can compute $log(-L)$ as above. Since $-I$ is a rotation by 180 degrees, we know that we would want to animate such a rotation by turning $\pm\pi t$ radians at time t. Hence, we use in this case $L(t) = R(\pm t\pi)e^{t\log(-A)}$, where $R(\alpha)$ is a rotation by angle α and the choice of sign corresponds to the two equally plausible interpolations of the 180 degrees rotation. Unfortunately, for matrices σ and τ which do not commute, $e^{\sigma+\tau} \neq e^{\sigma}\, e^{\tau}$. Therefore, this solution is not steady; but it meshes in well with the other cases (`case04` and `case07`).

3.6 The 3D case

The explicit computation of A^t in two dimensions hinges on having simple closed-form expressions for the eigenvalues and eigenvectors of 2×2 matrices. We have attempted first to attack the three dimensional problem using the same strategy, but quickly realized that the three-dimensional version is significantly more complicated. The eigenvalues are now the roots of a cubic, and hence using their algebraic expression to find a closed form solution for the exponential appears to be beyond reach. Consequently, the three-dimensional EAR solution proposed here is based on a different approach.

First, let us discuss how we transform the EAR problem to one of extracting the root of a linear part L (made up of the first three columns of A, $L = [IJK]$). We distinguish three cases, based on the number of fixed points of A. This is analogous to the two-dimensional cases discussed in connection with the algorithm in section 3.3, and has been outlined for three dimensions in the algorithm of section 3.4.

Computing the fixed points of A amounts to solving the system of three linear equations $AQ = Q$ or equivalently the homogeneous system $(L - \mathbb{1})Q = -T$. Therefore the existence and uniqueness of solutions is connected with whether 1 is an eigenvalue of L or not, and with the relationships between the eigenspaces of eigenvalue 1 and the non-homogeneous part of the system (given by the translation part T of the affinity):

Case A: When 1 is not an eigenvalue of L, then the affinity $A\,v = L\,v + T$ has a unique fixed point Q, since the determinant of $L - \mathbb{1}$ is not zero. In that case, instead of applying A^t to compute the transformed version $P_t = A^t P_0$ of a point P_0, we can apply L^t to compute the transformed version $QP_t = L^t QP_0$ of vector QP_0 and obtain P_t as $Q + QP_t$. Hence, the only thing remaining is computing L^t (`case14()`, `case15()`, `case16()` and `case18()`).

When 1 is an eigenvalue of L, with corresponding eigenvector v_1, we need to consider the translation part T of the affinity. We are looking for a solution of $AQ = Q$, or

—in terms of the linear part L and the translation part T— a solution of $(L-\mathbb{1})Q = -T$. In this case $\mathbb{R}^3$ decomposes in two subspaces $N = Ker(L-\mathbb{1})$ (the space of vectors whose image by $L-\mathbb{1}$ is the zero vector) and N^C (which satisfies that $(L-\mathbb{1})(N^C) \subseteq N^C$), so that $T = T_N + T_{N^C}$. We therefore have two more cases (handled by `case11()`, `case12()`, `case13()` and `case17()` in algorithm 4):

Case B: 1 is an eigenvalue of L and $T_N = \overrightarrow{0}$, in which case the system $(L-\mathbb{1})Q = -T$ is compatible, and has infinitely many solutions (since the $Ker(L-\mathbb{1})$ is non-trivial). We may pick any such solution as a fixed point (they all are), and again the affinity on P can be seen as a linear function acting on $\overrightarrow{QP}$, and the result being independent of our choice of fixed point Q.

Case C: 1 is an eigenvalue of L and $T_N \neq \overrightarrow{0}$. In this case, there is no solution of the system, and there is no fixed point. Instead, there is a straight line of points that maps onto itself. The composition of the affinity with a projection in the direction of T_N gives an affinity in the space N^C which does have a fixed point, and the line through that point in the direction of the vector T_N is the fixed line for the affinity. The motion we seek in this case is a generalized screw, which we call a swirl and define as a two-dimensional SAM on N^C combined with a translation along T_N. The translation amount is a linear function of time. Again we have to compute L^t in the lower dimension subspace, and add a linearly-varying contribution from T_N.

Now, let us focus on computing L^t. We distinguish two cases: a special case and then a general case.

Consider first the special case when the matrix L has three real eigenvalues. We can choose the corresponding (possibly generalized) eigenvectors as a basis and obtain the triangular version T of L in that basis, where $L = MTM^{-1}$. We split the problem into an exponential scaling (a scaling by λ_1^t, where λ is the first eigenvalue) in the direction of the eigenvector v_1, and solve for SAM in two dimensions on the plane spanned by the other generalized eigenvectors through the fixed point.

When the matrix has only one real eigenvalue, the procedure is the same, where this unique real eigenvalue and its eigenvector play the role of v_1 in the previous paragraph.

In our implementation, in dealing with the cases with exactly two or three eigenvalues equal to 1, we actually use special expressions of the exponential of a matrix to simplify the computation. These originate in the fact that

$$exp\left(t\begin{bmatrix} a & 0 & 0 \\ 0 & 0 & b \\ 0 & 0 & 0 \end{bmatrix}\right) = \begin{bmatrix} e^t a & 0 & 0 \\ 0 & 1 & tb \\ 0 & 0 & 1 \end{bmatrix},$$

and

$$exp\left(t\begin{bmatrix} 0 & a & b \\ 0 & 0 & c \\ 0 & 0 & 0 \end{bmatrix}\right) = \begin{bmatrix} 1 & ta & \frac{t^2 ac + tb}{2} \\ 0 & 1 & tc \\ 0 & 0 & 1 \end{bmatrix},$$

so that in these cases one easily obtains expressions for the logarithm of the transformation (in the basis of the generalized eigenvectors).

In the case where exactly one eigenvalue is 1, we proceed as in the case where there is only one real eigenvalue, choosing the eigenvector of eigenvalue 1 as the spiraling axis (although in this case there is no motion in that direction).

Notice that, when we carry out computations in the basis given by the (possibly generalized) eigenvectors, and since the matrix need not be symmetric, these eigenvectors are not necessarily orthogonal, and the matrix of eigenvectors may be ill conditioned. Nonetheless, our reference implementation was able to compute a solution even for very large condition numbers (in the order or 10^{15}). These extreme cases, however, where purposely constructed for testing, and are hard to come upon when interacting with the application directly.

4 Existence of a real root

In the previous section, we have discussed situations where a SAM exists and can be computed using the EAR algorithm proposed in the Section 3. Several important questions remain, considering that a real $\log(A)$ may not exist:

1. Can we detect the bad configurations where EAR fails because there is no real matrix solution for $log(A)$ and hence for A^t?
2. Can a SAM exist in such a bad configuration?
3. Can we define a safe set of good configurations that is sufficiently large to be useful for common applications?
4. What motions should we use in the bad configurations?

We answer the fist two questions here and offer observations and conjectures for the other two that are based on theoretical observations and thorough experiments. A formal treatment of these last two questions is still in progress.

4.1 Characterization of bad configurations

To compute the SAM, we need to compute L^t, which, depending on the case, can be done either through diagonalization or by computing $\log(L)$. Even when it is done through diagonalization, the conditions that it needs to meet ensure the existence of a real $\log(L)$. A precise characterization of when this happens is given in [Cul66]: the algebraic and geometrical multiplicities of each real negative eigenvalue of L must be equal and even. Notice that this condition may not be satisfied by arbitrary orientation preserving affinities, contrary to what is stated in [ACOL00], as we discuss in the following subsection. However, all orientation preserving affinities with a homogeneous scaling (similarities) do have a real logarithm. Since we are interested here in dimensions 2 and 3, this characterization means that if there is a negative eigenvalue, it must be a double eigenvalue with two linearly independent eigenvectors. In this case the affinity is a rotation by 180 degrees (possibly combined with a uniform scaling). Then both eigenvalues are $-s$, where s is the scale factor. However, affinities arbitrarily "near" this one may have distinct negative eigenvalues, and hence may not have a real logarithm.

4.2 Geometry of configurations where SAM exists

The existence condition quoted in the previous Subection does not provide a geometric picture of the valid configurations. We discuss here its implications in the case of three dimensions. To explore the nine-dimensional space of configurations (the coefficients of L) and gain insight as to the subspace where EAR successfully computes A^t, we have decided to display color-coded samples on a user-controlled slice of that configuration. We use a control tetrahedron (A_0, B_0, C_0, D_0) and its image (A_1, B_1, C_1, D_1) by the affinity A. Hence, we manipulate A by moving these vertices. To visualize a particular slice, we keep the two control tetrahedra constant, except for D_1 (shown in magenta). We place D_1 on the points of a regular grid on the plane P passing through the original position of D_1 and parallel to the triangle (A_1, B_1, C_1). The sampling is centered around the closest projection on P of the barycenter of triangle (A_1, B_1, C_1). The size of the sampled square is proportional to the area of triangle (A_1, B_1, C_1). Note that the samples depend on the distance between D_1 and the plane passing through triangle (A_1, B_1, C_1), but not on the position of D_1 on P. We then temporarily place D_1 at each one of these samples and display a small sphere around it, which we paint green when EAR succeeded to compute a SAM and red when it failed. Hence, the set or red spheres depict a slice through the bad region of this nine-dimensional configuration space. Using this set-up and interactive tools for translating, scaling, rotating the two control tetrahedral and for moving their individual vertices, we have made the following observations.

Observation 1: For a range of relative orientations that does not approach a 180 degree turn and for a useful range of relative distortions (produced by dragging individual markers away from their initial or rotated position), EAR successfully produces a SAM and in fact keeps all samples green (Figure 6).

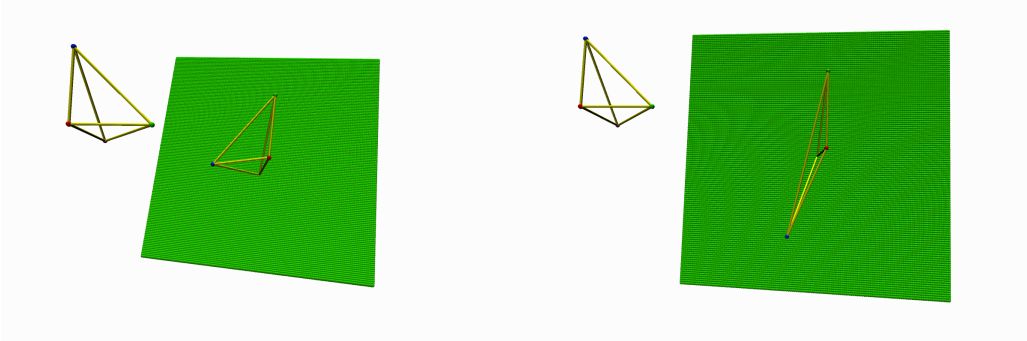

Figure 6: Two different configurations, and the sample our tool shows of good and bad positions for the purple vertex (in this cases, all are good).

Observation 2: The good (green) region is convex (Figure 7). We have looked at hundreds of examples obtained through interactive exploration and have never witnessed a non-convex green area although we do not know of a formal proof of this fact.

Observation 3: In a configuration where A is a rigid body motion close to a 180 degree rotation around an arbitrary axis, EAR produces the correct screw motion, but the screw is at the tip of a sharp end of the good (green) region (Figure 8). A slight

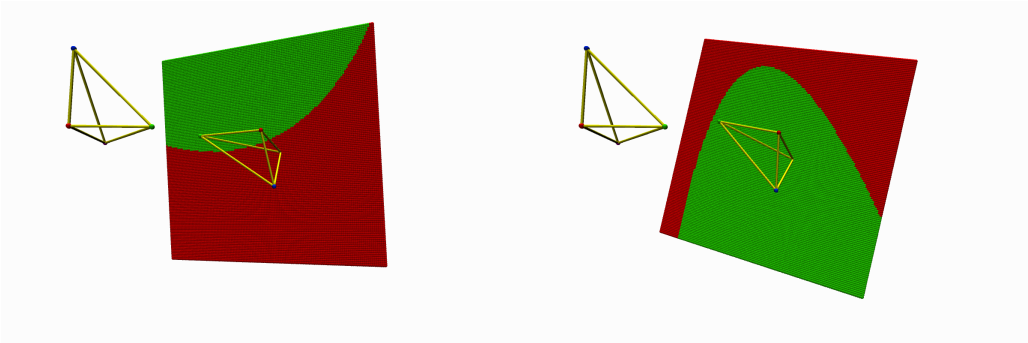

Figure 7: The good (green) region appears convex in all samples we've explored.

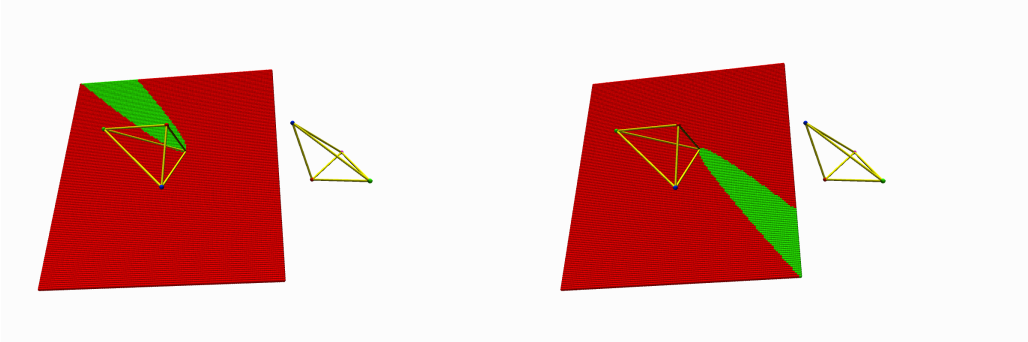

Figure 8: The good region narrows near the extreme case of rotations by 180 degrees.

perturbation of the angle past 180 degrees flips the green region to the other side of the screw motion (classical elbow up/elbow down singularity). Hence, slight perturbations of the markers away from a rigid body motion consisting of such a turn by 180 degrees have a good chance of producing invalid configurations for which we cannot find a SAM.

Observation 4: The coloring of the slice is not affected by relative translation and relative scaling between the two control tetrahedra (see Figure 9).

Since we have not been able to characterize formally the geometry of the region of good configurations, we have written a program to perform an exhaustive search over a limited region in the space of affinities, in an attempt to identify an easy-to-use "safe" subset of the good configurations where we can guarantee EAR will succeed.

Since a scaling will not affect the existence of a solution, our sample space consists of a set of directions, a set of rotations around those directions, and a set of perturbations of each element in the resulting matrix. To sample directions, we've constructed an icosahedron, subdivided it 2 times, and tested all axes emanating from the origin and passing through a vertex of the result. We then tested rotations around those axes from $-\theta$ to θ, and for each such rotation, we've added a perturbation to each element of the matrix by $-d$, 0 or d in all possible combinations, and checked if the resulting matrix had

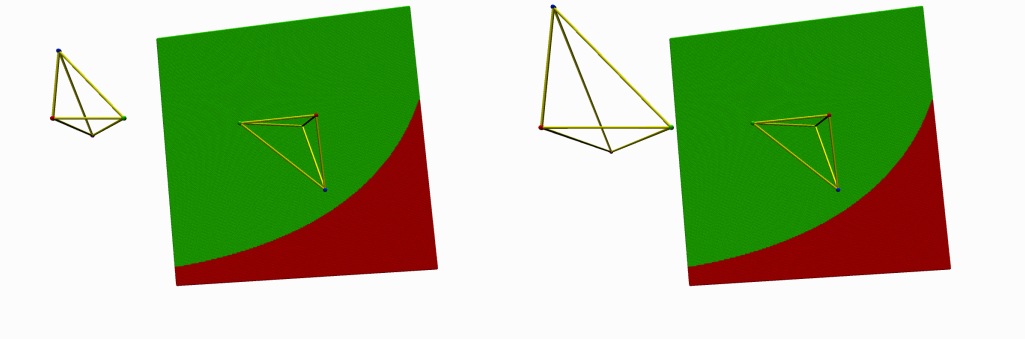

Figure 9: The shape of the boundary between good (green) and bad (red) positions for vertex D_1 does not depend on relative position or scale.

a logarithm. Orientation-inverting matrices where silently discarded.

We have found that for $\theta = 45$ degrees, and $d = .5$, all the resulting transformations are good (to assess the significance of d, keep in mind that the unperturbed matrices are orthogonal, and therefore all columns have norm 1). Similarly, if we increase the rotation angle to 90 degrees, but reduce the size of the perturbations to $d = 0.33$, all configurations on our dense grid are good. An even larger rotation by 120 degrees requires further reducing the perturbation amount to $d = 0.29$. These experiments seem to confirm our expectation that the region where SAM exists is rather large, and it becomes narrower as we approach a rotation by 180 degrees. Notice that these characterization bounds represent conservative estimates, that guarantee that no ill configuration will be found in those regions. The domain of SAM, however, extends well beyond those bounds.

5 Properties and Applications of SAM

Steady affine motions have several beautiful properties. In this section, we mention only a few, hoping to motivate further exploration.

First, let us answer the following question. Given an affinity A, can there be more than one steady affine motion that interpolates between the identity $\mathbb{1}$ and A?

5.1 Uniqueness

For a time-parameterized affinity A_t, the property that the speed of all points remains constant in the local frame (given by the columns of L) is equivalent to an ordinary differential equation on the entries of the affinity that has the form

$$\frac{d}{dt}L(t) = M\,L(t), \tag{13}$$

and

$$\frac{d}{dt}T(t) = L(t)\begin{bmatrix}\alpha\\ \beta\end{bmatrix} \tag{14}$$

where M is a matrix of constants related with the speed of each point in local coordinates, and α and β are constraints determined by the translation part T. This is the reason why we have defined a SAM the way we have: the solutions of this equation are of the form $A(t) = e^{tM}$ [Arn81], and since we want $A(1) = L$ (our scaling convention), M must be $log(L)$. It then follows from the uniqueness of the solutions to this differential equations that there can only be one SAM between two given affinities if we fix the times (in our case 0 and 1) when the motion interpolates the given affinities.

Notice also that by choosing the origin at the fixed point of the affinity A, the translation part is zero, and we can therefore choose $\alpha = \beta = 0$. This works in all cases safe some singular cases (when L has an eigenvalue 1), which are the only cases in which one needs to deal with these constants (see Section 3.5).

Furthermore, when L is a matrix that has no real logarithm, if there existed a SAM $S(t)$ interpolating F_0 and $F_n = [L, T]\, F_0$, that motion would need to satisfy the previous differential equation. Then by the uniqueness of the solutions to the differential equation, we would have found the logarithm of L nonetheless. Therefore, we conclude that when L has no real logarithm, it is impossible to find a steady motion that interpolates A.

5.2 Constant velocity

Let us now investigate the instantaneous velocity of a point P subject to a SAM. In the local frame, according to the discussion in the previous subsection concerning equations (13) and (14), a SAM moves every point with constant speed in the local frame, which may be computed using the matrix $M = \log(L)$. In fact, in the local frame, the velocity vector of a point that starts at (x_0, y_0) at time $t = 0$ is given by $M^+ \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$ at any point in time (where M^+ denotes the transpose of M).

Moreover, the speed vector field is stationary (i.e. the speed at each point in the plane is constant, regardless of the value of the time parameter t at which it is visited). Indeed, in the general case, where 1 is not an eigenvalue of L, our choice of origin makes $T(t) \equiv 0$, and we see that the velocity at the point Q is $M\,Q$, by considering any trajectory that visits Q at some given time t:

$$Q = P(t) = A_t\, P = L(t)\, P = e^{tM}\, P,$$

we see by differentiating that the speed of $P(t)$ is

$$P'(t) = M\, e^{tM}\, P = M\, e^{tM}\, P.$$

The fact that we've chosen a special origin that makes $T \equiv 0$ is not essential, but just a convenience to make the computation simpler. In the cases where L does have some eigenvalue equal to one, the expression for $T(t)$ is linear in t, and therefore the statement is also true in those cases.

5.3 Preservation of properties

Here we discuss quantities such that, if A preserves them, then also a SAM A_t preserves them for every time t.

5.3.1 Isometry (distance and angles)

A is an isometry if L is an orthogonal matrix (here, with positive determinant, therefore equal to one). In this case it is either the identity, or it is a rotation (with a possible translation along its axis in the three dimensional case). The corresponding SAM will then be a constant-speed rotation (plus possibly a constant speed translation), and therefore an isometry.

5.3.2 Similarity (angles)

A similarity is a composition of a rotation and an homogeneous scaling. The corresponding SAM will result from the composition of a constant speed rotation and an exponential homogeneous scaling, and is therefore also a similarity at every point in time.

5.3.3 Area (in 2D) and Volume (in 3D)

For an affinity A to preserve volume (area in the case of an affinity in the plane), the necessary and sufficient condition is that $det(L) = 1$. Since for matrix exponentiation $\det(e^B) = e^{\mathrm{trace}B}$, we conclude that for a volume-preserving affinity A, $\mathrm{trace}(\log(L)) = 0$, so $\mathrm{trace}(t\log(L)) = t \cdot \mathrm{trace}(\log(L)) = 0$, and hence $\det(L_t) = \det(e^{t\cdot\log(L)}) = e^{\mathrm{trace}(t\cdot\log(L))} = 1$.

5.4 Monotonic variation of volume (area)

When A does not preserve volume, then A^t evolves it monotonically. As in (5.3.3) above, we notice that $\mathrm{trace}(\log(L)) = \log(\det(L))$ will be positive or negative depending on whether A increases or decreases volume. Then $\frac{d}{dt}\det(L(t)) = \mathrm{trace}(\log(L)e^{t\cdot\mathrm{trace}(\log(L))}$ has constant sign, and therefore the variation of volume is monotonic.

5.5 Sweeps

The envelope swept by a moving surface S_t is the union of the grazing points [AMBJ06] of S_t, that is those points P of S_t whose instantaneous velocity is tangential to S_t. For general motions, the set of grazing points evolves on S_t with time. When the motion is approximated by a set of screw motions, these grazing points remain constant during each screw motion. This property was exploited in [RKS+07]. We extend it here to SAMs. For example, a characteristic point $Q + s\vec{T}$ along a line through Q with tangent $\vec{T}$ on a plane with normal $\vec{N}$ satisfies $\vec{N} \cdot (\log(L)(\overrightarrow{FQ} + s\vec{T})) = 0$ yielding

$$s = \frac{\vec{N} \cdot \log(L)(\overrightarrow{FQ})}{\vec{N} \cdot \log(L)(\vec{T})}.$$

For example, for a triangle moving under the action of a SAM, we may use the previous formula to find the 0, 1, 2, or 3 grazing points on its boundary. Tracing those points under the SAM will then yield the envelope of the triangle under that motion.

6 Conclusions

An affine motion A_t that interpolates between the identity $A_0 = \mathbb{1}$ and an affinity $A_1 = A$, is often needed to generate a continuous animation or a discrete set of evenly sampled frames that define a regular pattern of geometric features. A variety of formulations have been proposed. The best choice may be dictated by subjective aesthetic criteria and by domain-dependent needs. In the absence of such criteria, when A is a rigid body transformation, a screw interpolation is often used because it combines a natural and pleasing motion with a set of useful geometric properties. In two dimensions, the screw motion reduces to a pure rotation or a pure translation. When A is no longer restricted to be rigid, but is a planar similarity, the aesthetic and computational benefits of screw motions may be extended by using a logarithmic spiral interpolation, which includes the golden spiral famous for its beauty and which was named "spira mirabilis" (the miraculous spiral) because of its surprising and beautiful properties.

The Steady Affine Motion (SAM) introduced here extends the screw solution to arbitrary affinities and the two-dimensional logarithmic spiral solution to situations where A is not restricted to a similarity and to three dimensions. An affine motion is steady when it satisfies $A_t = A^t$, which we name the "equation of beautiful motions".

SAMs have several interesting properties. For example, (1) the affine relation between evenly spaced frames of a SAM is constant and (2) the instantaneous velocity of a point remains constant over time both in the global and the local (moving) frame. These properties result in pleasing curved trajectories of points and lower algebraic complexity when computing derived entities, such as the envelope of a swept volume.

The SAM solution proposed here differs from previously proposed approaches. For example, because SAM is steady and the previously proposed "as-rigid-as-possible" shape-interpolation [ACOL00] is not; and even if one ignores absolute position, SAM produces shape interpolations that are different from the as-rigid-as-possible ones. Furthermore, the solution proposed by Alexa [Ale02] which expresses M_t as $\exp((1-t)\log(A)+t\log(B))$ is also not steady, and therefore does not enjoy the properties stated above, while our solution, $\exp(t\log(BA^{-1}))A$, does.

We provide the implementation details and theoretical justification of our Extraction of Affinity Roots (EAR) algorithm, which uses closed form expressions to compute A^t in two and three dimensions, when it exists, and hence eliminates the performance, accuracy and stability shortcomings of previously proposed dimension-independent numeric solutions.

We demonstrate the robustness of our solution using a direct manipulation environment, where the designer may edit the starting frames and ending frames of two SAMs, A and B, and where we automatically compute and display the intermediate frames of SAMs that each interpolate a different pair of corresponding intermediate frames A_i and B_i. Hence letting the user interactively control a smooth bi-variate steady pattern of frames.

We show that the solution produced by EAR is the only solution satisfying the equation of beautiful motions. When no SAM solution exists, we propose an unsteady alternative that naturally extends the steady solutions by combining them with a linear interpolation of a rotation by 180 degrees. Notice however that all similarities do have a SAM. Furthermore, we conjecture and verify experimentally that SAM solutions exist

for a large and dense set of affinities derived from the identity by an arbitrary translation and global scaling, but with interdependent bounds on the amount of rotation and shear. For example, we believe that a SAM solution always exist when the total rotation angle is less than 90 degrees and the shear (after the global scaling has been factored out) is a 3×3 matrix whose coefficients are obtained by perturbing the coefficients of the identity matrix by no more than 0.33. The bound on the shear perturbation decreases as the bound on the rotation angle is increased.

We believe that the simplicity and beauty of the Steady Affine Motion makes it a prime candidate for designing motions and patterns and for representing local (instantaneous) approximations of more general affine motions just as screw motions are used to locally approximate rigid motions in three dimensions.

References

[ACOL00] Marc Alexa, Daniel Cohen-Or, and David Levin. As-rigid-as-possible shape interpolation. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 157–164, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[AIS77] C. Alexander, S. Ishikawa, and M. Silverstein. *A Pattern Language: Towns, Buildings, Construction.* Oxford University Press, New York, 1977.

[Ale02] Marc Alexa. Linear combination of transformations. In John Hughes, editor, *SIGGRAPH 2002 Conference Proceedings*, Annual Conference Series, pages 380–387. ACM Press/ACM SIGGRAPH, 2002.

[AMBJ06] Karim Abdel-Malek, Denis Blackmore, and Kenneth Joy. Swept volumes: Foundations, perspectives, and applications. *International Journal of Shape Modeling*, 12(1):87–127, June 2006.

[Arn81] V. I. Arnold. *Ordinary Differential Equations.* The MIT Press, 1981.

[BCGH92] Alan H. Barr, Bena Currin, Steven Gabriel, and John F. Hughes. Smooth interpolation of orientations with angular velocity constraints using quaternions. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 313–320. ACM Press, 1992.

[BF01a] Buss and Fillmore. Spherical averages and applications to spherical splines and interpolation. *ACMTG: ACM Transactions on Graphics*, 20, 2001.

[BF01b] Samuel R. Buss and Jay P. Fillmore. Spherical averages and applications to spherical splines and interpolation. *ACM Trans. Graph.*, 20(2):95–126, 2001.

[BLCD02] Christoph Bregler, Lorie Loeb, Erika Chuang, and Hrishi Deshpande. Turning to the masters: motion capturing cartoons. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 399–407, New York, NY, USA, 2002. ACM.

[CHKL00] Sheung Hun Cheng, Nicholas J. Higham, Charles S. Kenney, and Alan J. Laub. Approximating the logarithm of a matrix to specified accuracy. *SIAM J. Matrix Anal. Appl.*, 22(4):1112–1125, 2000.

[Cli82] W. Clifford. *Mathematical Papers*. Macmillan, London, 1882.

[CS03] Jaeil Choi and Andrzej Szymczak. On coherent rotation angles for as-rigid-as-possible shape interpolation. Technical report, Georgia Institute of Technology, 2003.

[Cul66] Walter C. Culver. On the existence and uniqueness of the real logarithm of a matrix. *Proceedings of the American Mathematical Society*, 17(5):1146–1151, October 1966.

[DH03] Phillip I. Davies and Nicholas J. Higham. A schur-parlett algorithm for computing matrix functions. *SIAM J. Matrix Anal. Appl*, 2003.

[Duf86] Tom Duff. Splines in animation and modeling. In *SIGGRAPH '86 Course Notes on State of the Art Image Synthesis*. ACM Press, 1986.

[Eat00] John W. (John Wesley) Eaton. *GNU Octave: a high-level interactive language for numerical computations: edition 3 for Octave version 2.0.13*. Network Theory Ltd., pub-NETWORK-THEORY:adr, 2000.

[FTA05] H. Fu, C. L. Tai, and O. K. Au. Morphing with laplacian coordinates and spatial-temporal texture. In *Pacific Graphics'05*, pages 100–102, 2005.

[GDCV98] Jonas Gomes, Lucia Darsa, Bruno Costa, and Luiz Velho. *Warping and morphing of graphical objects*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998.

[Gra98] F. Sebastian Grassia. Practical parameterization of rotations using the exponential map. *Journal of Graphics Tools: JGT*, 3(3):29–48, 1998.

[GV96] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, third edition, 1996.

[HH05] Desmond J. Higham and Nicholas J. Higham. *MATLAB Guide*. SIAM, 2005.

[Hig86] Nicholas J. Higham. Newton's method for the matrix square root. *Math. of Comp.*, 1986.

[HJK02] Dae-Eun Hyun, Bert Jüttler, and Myung-Soo Kim. Minimizing the distortion of affine spline motions. *Graphical Models*, 2002.

[JR09] Justin Jang and Jarek Rossignac. Octor: Subset selection in recursive pattern hierarchies. *Graphical Models*, In Press, Corrected Proof:–, 2009.

[JT95] Ollie Johnston and Frank Thomas. *The Illusion of Life: Disney Animation*. Disney Press, October 1995.

[KCŽO08] Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O'Sullivan. Geometric skinning with approximate dual quaternion blending. *ACM Transactions on Graphics*, 27(4):105:1–105:??, October 2008.

[KKS95] Myoung-Jun Kim, Myung-Soo Kim, and Sung Yong Shin. A general construction scheme for unit quaternion curves with simple high order derivatives. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 369–376. ACM Press, 1995.

[KN95] M. S. Kim and K. W. Nam. Interpolating solid orientations with circular blending quaternion curves. *Computer Aided Design*, 27:385–398, 1995.

[KR03] ByungMoon Kim and Jarek Rossignac. Collision prediction. *J. Comput. Inf. Sci. Eng*, 3(4):295–301, 2003.

[Kv05] Ladislav Kavan and Jiří Žára. Spherical blend skinning: a real-time deformation of articulated models. In *I3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 9–16, New York, NY, USA, 2005. ACM.

[LG06] Mathieu Larive and Veronique Gaildrat. Wall grammar for building generation. In *GRAPHITE '06: Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*, pages 429–437, New York, NY, USA, 2006. ACM.

[LKG+03] Ignacio Llamas, Byungmoon Kim, Joshua Gargus, Jarek Rossignac, and Chris D. Shaw. Twister: a space-warp operator for the two-handed editing of 3D shapes. In Jessica Hodgins and John C. Hart, editors, *Proceedings of ACM SIGGRAPH 2003*, volume 22(3) of *ACM Transactions on Graphics*, pages 663–668, 2003.

[LS02] J. Lee and S. Y. Shin. General construction of time-domain filters for orientation data. In *IEEE Transcations on Visualization and Computer Graphics*, volume 8(2), pages 119–128. IEEE Computer Society, 2002.

[MCJ00] Lizhuang Ma, Tony K. Y. Chan, and Zhongding Jiang. Interpolating and approximating moving frames using B-splines. In *Pacific Conference on Computer Graphics and Applications*, pages 154–164. IEEE Computer Society, 2000.

[Mei05] Beatrice Meini. The matrix square root from a new functional perspective: Theoretical results and computational issues. *SIAM J. Matrix Anal. Appl.*, 26(2):362–376, 2005.

[Mor07] Michael Mortenson. *Geometric Transformations for 3D Modeling*. Industrial Press Inc, New York, 2007.

[PBPS99] Joanna L. Power, A. J. Bernheim Brush, Przemyslaw Prusinkiewicz, and David H. Salesin. Interactive arrangement of botanical l-system models. In *I3D '99: Proceedings of the 1999 symposium on Interactive 3D graphics*, pages 175–182, New York, NY, USA, 1999. ACM.

[RBG88] K. S. Roberts, G. Bishop, and S. K. Ganapathy. Smooth interpolation of rotational matrices. In *Proceedings CVPR '88: Computer Vision and Pattern Recognition*, pages 724–729. IEEE Computer Science Press, 1988.

[RK01] Jarek Rossignac and Jay J. Kim. Computing and visualizing pose-interpolating 3D motions. *Computer-Aided Design*, 33(4):279–291, 2001.

[RKS+07] Jarek Rossignac, J. J. Kim, S. C. Song, K. C. Suh, and C. B. Joung. Boundary of the volume swept by a free-form solid in screw motion. *Computer-Aided Design*, 39(9):745–755, 2007.

[SD92] Ken Shoemake and Tom Duff. Matrix animation and polar decomposition. In *Proceedings of Graphics interface '92*, pages 258–264. Morgan Kaufmann Publishers Inc., 1992.

[Sho85] Ken Shoemake. Animating rotation with quaternion curves. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 245–254. ACM Press, 1985.

[Sho87] Ken Shoemake. Quaternion calculus and fast animation. In *SIGGRAPH '87 Course Notes on State of the Art Image Synthesis*, pages 101–121. ACM Press, 1987.

[vERR93] Maarten van Emmerik, Ari Rappoport, and Jarek Rossignac. Simplifying interactive design of solid models: A hypertext approach. *The Visual Computer*, 9(5):239–254, March 1993.

[WJ93] Wenping Wang and Barry Joe. Orientation interpolation in quaternion space using spherical biarcs. In *Graphics Interface '93*, pages 24–32, May 1993.

[WWSR03] Peter Wonka, Michael Wimmer, François Sillion, and William Ribarsky. Instant architecture. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 669–677, New York, NY, USA, 2003. ACM.